processor. A higher ~~Maximum~~ instruction-level parallelism for the loop can be [[is]] realized if the recurrence initiation interval of the loop is less than or equal to its resource initiation interval. Typically, this condition is not satisfied for loops whose computations involve sparse arrays/matrices. The body of such a loop typically starts with a load whose address in itself is an element of another array (called the index array) and ends with a store whose address is an element of the index array. In the absence of static information, the compiler does not know about the contents of the elements of the index array. Hence, it must assume that there is a loop-carried dependence edge from the store in one iteration to the load in the next iteration. This makes the recurrence initiation interval much higher than resource initiation interval.

The paragraph beginning at page 5, line 17, is amended as follows:

The following example illustrates the problem of recurrence initiation interval being higher than resource initiation interval in a vectorizable loop requiring computation of sparse arrays/matrices:

First consider a loop performing computations on arrays that are directly accessed:

The paragraph beginning at page 5, line ~~16~~ 26, is amended as follows:

SVP 5.8.07 10/612,724

Now consider the ~~The~~ following loop, which represents sparse "Gather Vector Add," illustrates a common operation in sparse matrix computations:

The paragraph beginning at page 6, line 10, is amended as follows:

Following is an example Intel® Itanium® architecture ~~the~~ code generated for the above loop, without software-pipelining:

The paragraph beginning at page 6, line 12, is amended as follows:

```
.b1_1:                          // 21 cycles per iteration
  { .mmi
      ld8    r9=[r32],8         // cycle 0:  load b[i]
      ldfs   f8=[r34],4         // cycle 0:  load c[i]
      nop.i 0 ;;
} { .mmi
      shladd r2=r9,2,r33 ;;     // cycle 2:  address of (a[b[i]])
      ldfs   f7=[r2]            // cycle 3:  load a[b[i]]
      nop.i 0 ;;
```